# Data Analytics on DBLP Dataset

Sponsor - Jia Zhang
Point of contact - Hoa Vu
Faculty Advisor - Jia Zhang
Team - 7 (Minghan Chen, Kun-Lin Lee, Johnny Shen, Hoa Vu)

Data Intensive Workflow Development for Software Engineers (18-656), Fall 2015

# Roadmap

- Introduction
- Motivation
- System design
- System implementation
- Demo
- Related work
- Experiments/analysis
- Conclusions and future work

# Introduction

Use a graph database to perform data storage, retrieval, and analytics, as well as provide a web portal interface to visualize results.

- Store dataset into Neo4j
- Execute queries using Cypher
- Present results with SpringBoot, D3



# Motivation

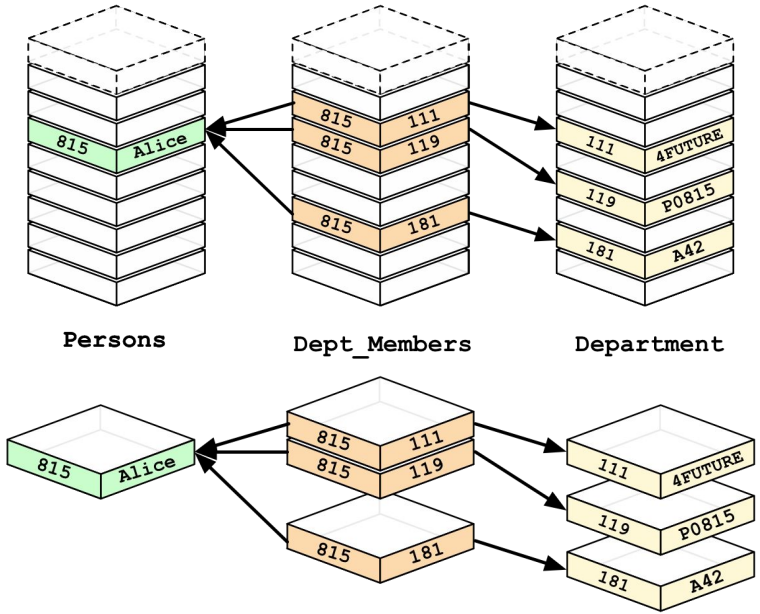
Increasingly relevant (Big Data, Analytics, Integration)

Graph Databases:

- Implicit relationships
- Flexible, intuitive models

Neo4j:

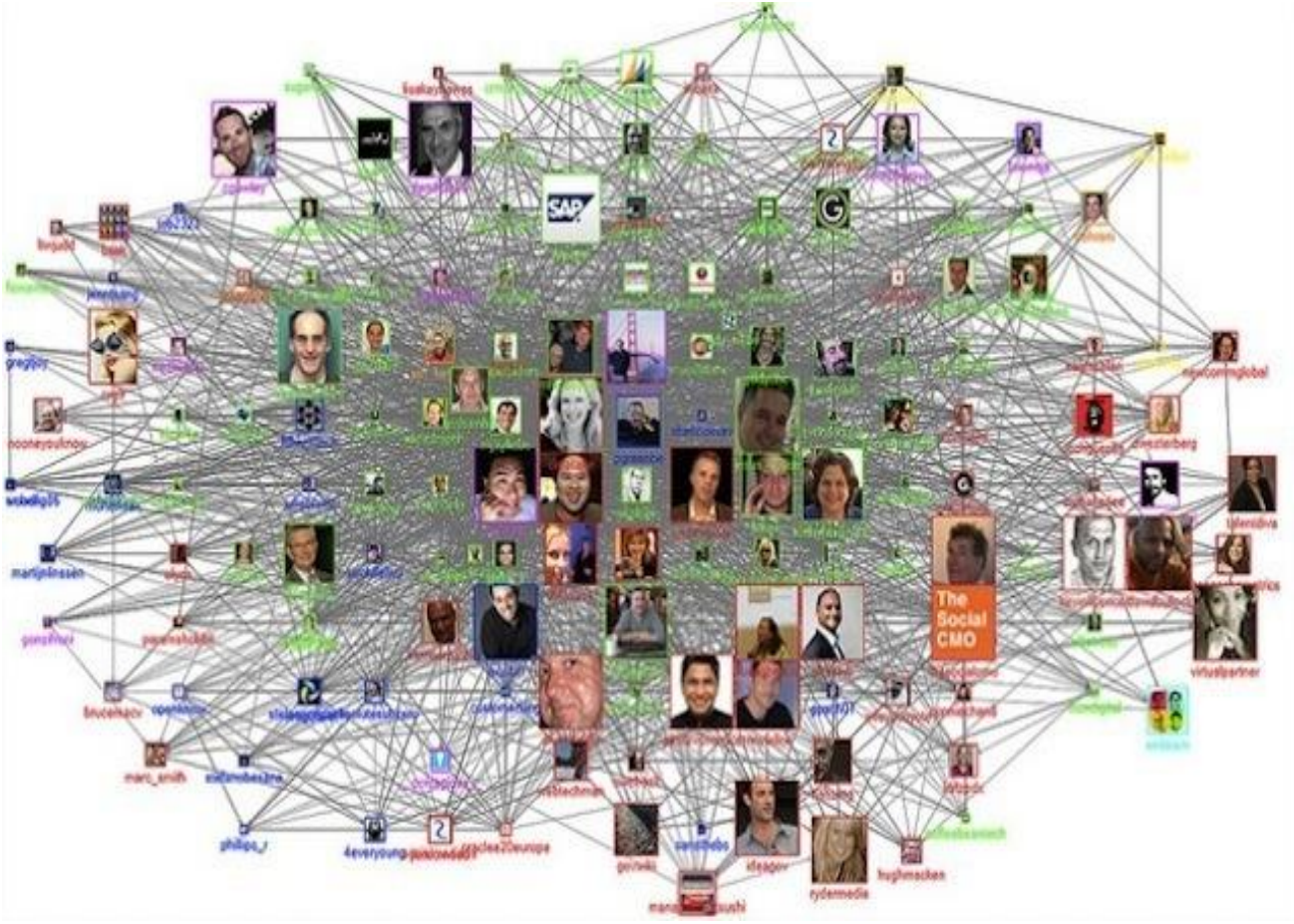
- High availability
- Performance, scalability



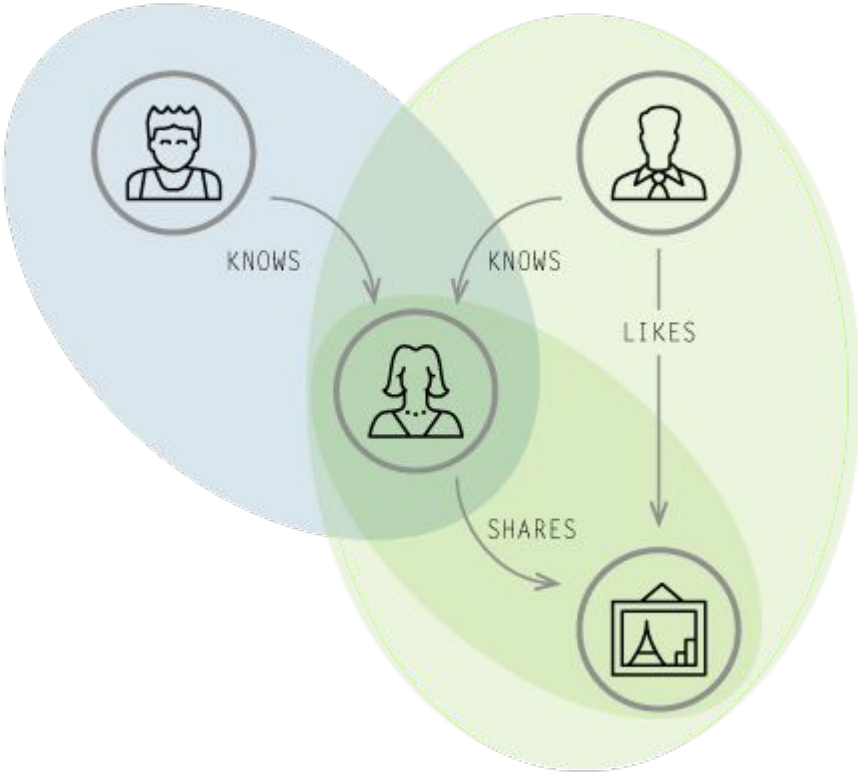

"From Relational to Neo4j", http://neo4j.com/developer/graph-db-vs-rdbms/

# Related Work - Google Knowledge Graph

# Social Network



# Related Work



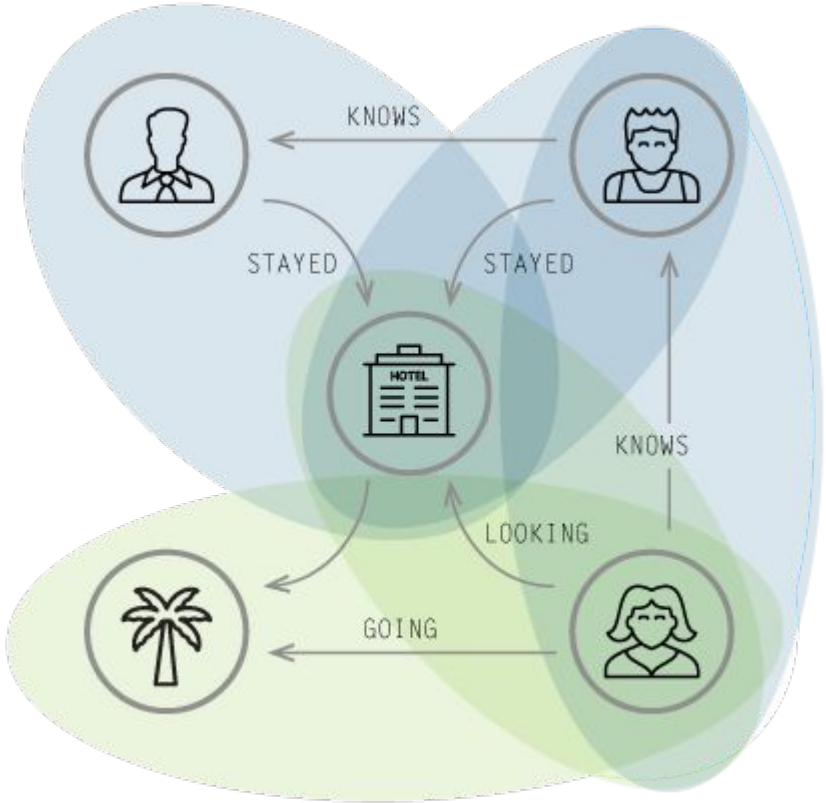

## Social Network

- Collaboration and sharing
- Friend of friend recommendations
- Discover unique relationships

# Related Work

## Real-Time Recommendation

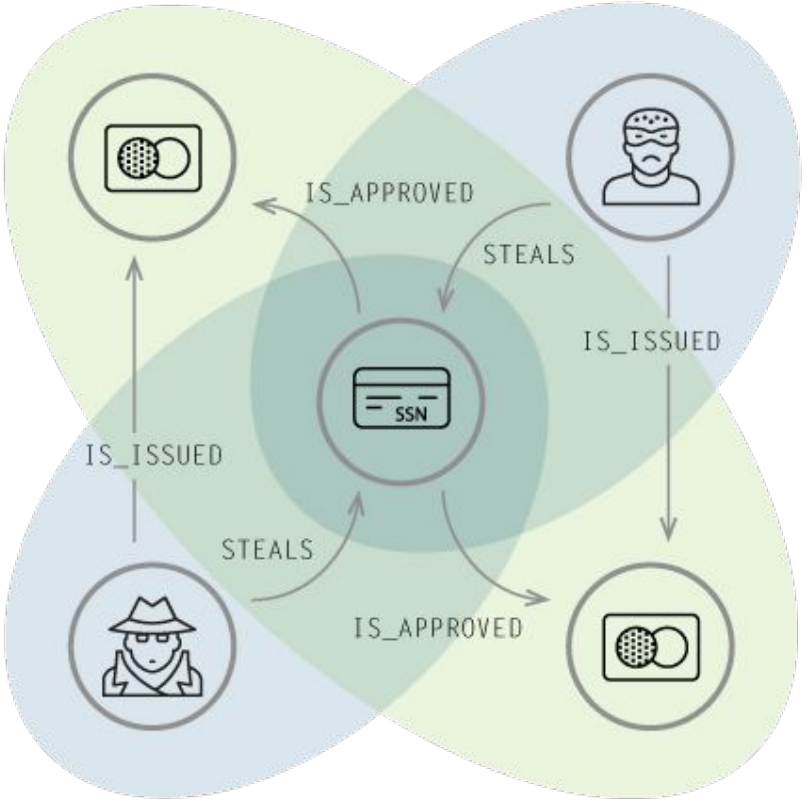
- Personalize user recommendations
- Multi criteria search



# Related Work

## Fraud Detection

- Catch fraud rings and prevent their damage by augmenting discrete data scrutiny with data relationship analysis



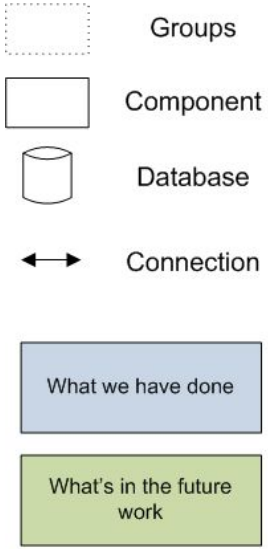
# Related Work

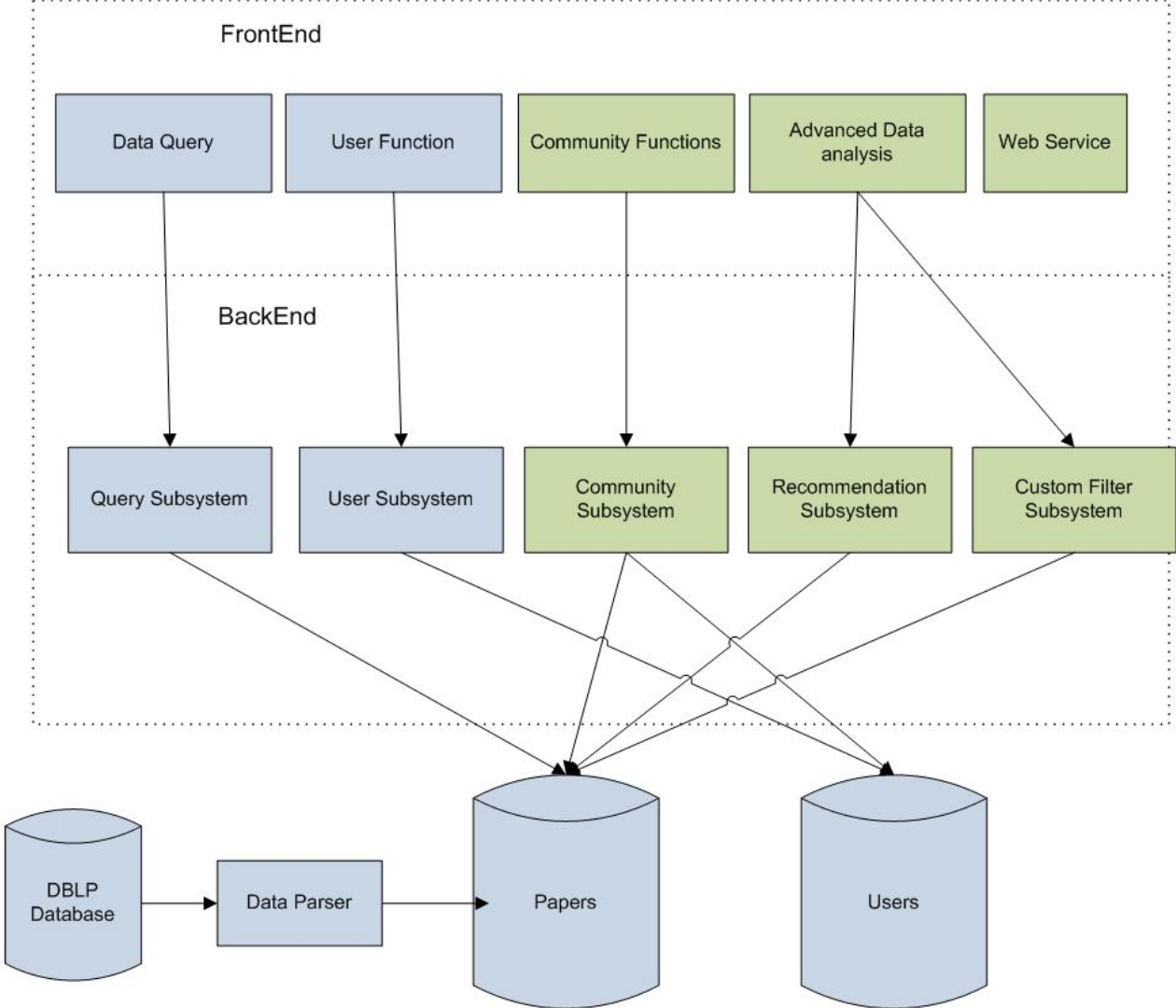
**Stanford Network Analysis Project using DBLP dataset**

- Constructed co-authorship network where two authors are connected
- Define and evaluate > 5000 network communities based on ground-truth


| Dataset statistics | |
|---|---|
| Nodes | 317080 |
| Edges | 1049866 |
| Nodes in largest WCC | 317080 (1.000) |
| Edges in largest WCC | 1049866 (1.000) |
| Nodes in largest SCC | 317080 (1.000) |
| Edges in largest SCC | 1049866 (1.000) |
| Average clustering coefficient | 0.6324 |
| Number of triangles | 2224385 |
| Fraction of closed triangles | 0.1283 |
| Diameter (longest shortest path) | 21 |
| 90-percentile effective diameter | 8 |

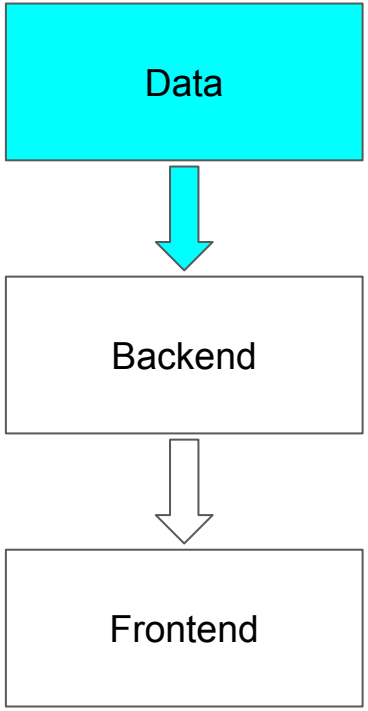
# System Design



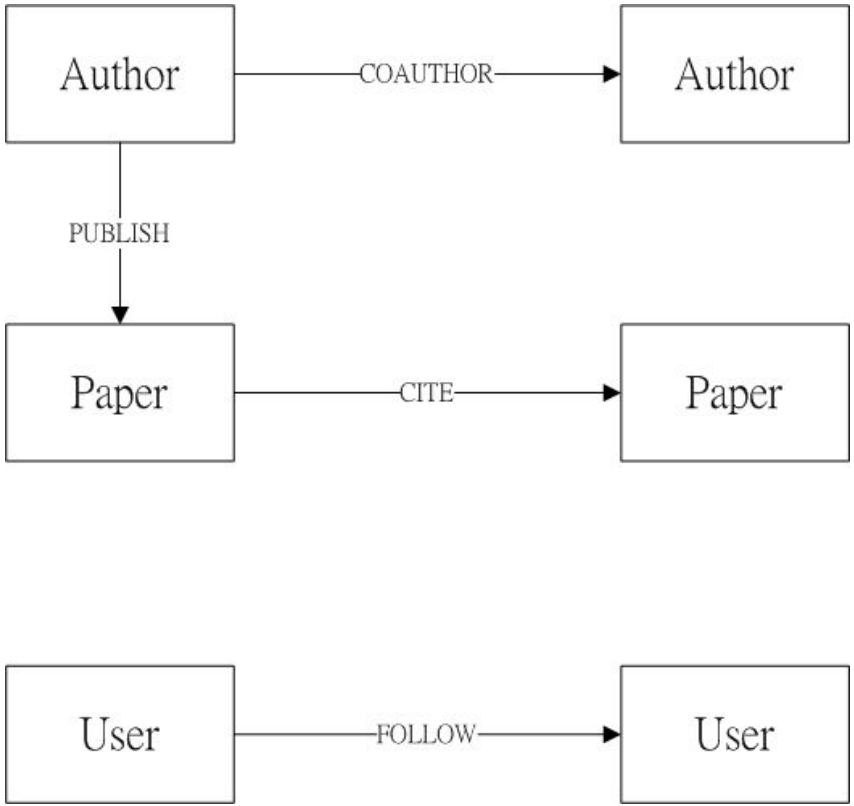



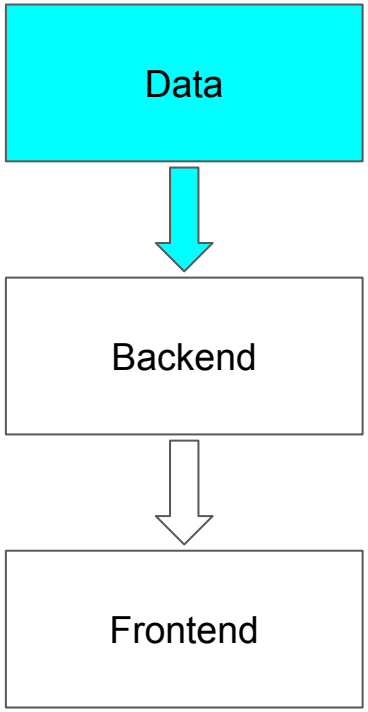
# System Implementation

- The publication data comes from dblp database. It's in XML format
- We wrote a python parser to transfer it into csv format and write it into database
- The paper data is stored in Neo4j. In order to take advantage of the graphical nature Neo4j, we added new nodes and relationtions like author node, coauthor relationship etc.
- User information is also stored in Neo4j



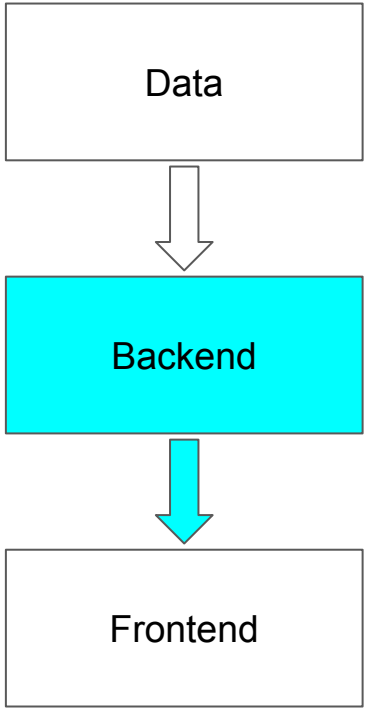
# System Implementation



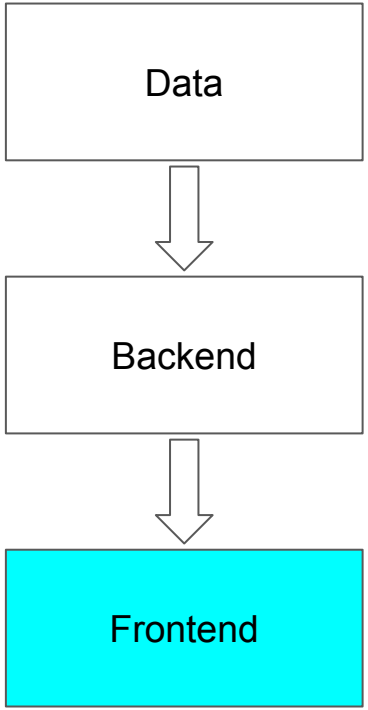
# System Implementation

- Our backend application is based on an example provided by TA Using Java and Spring Boot.
- Spring Security is introduced for user functions. Users without an account can’t use the system.
- Spring Data Neo4j is used to communicate with neo4j database.
- All dependencies are managed by Maven.
- An MVC structure is added to handle dynamic page contents.

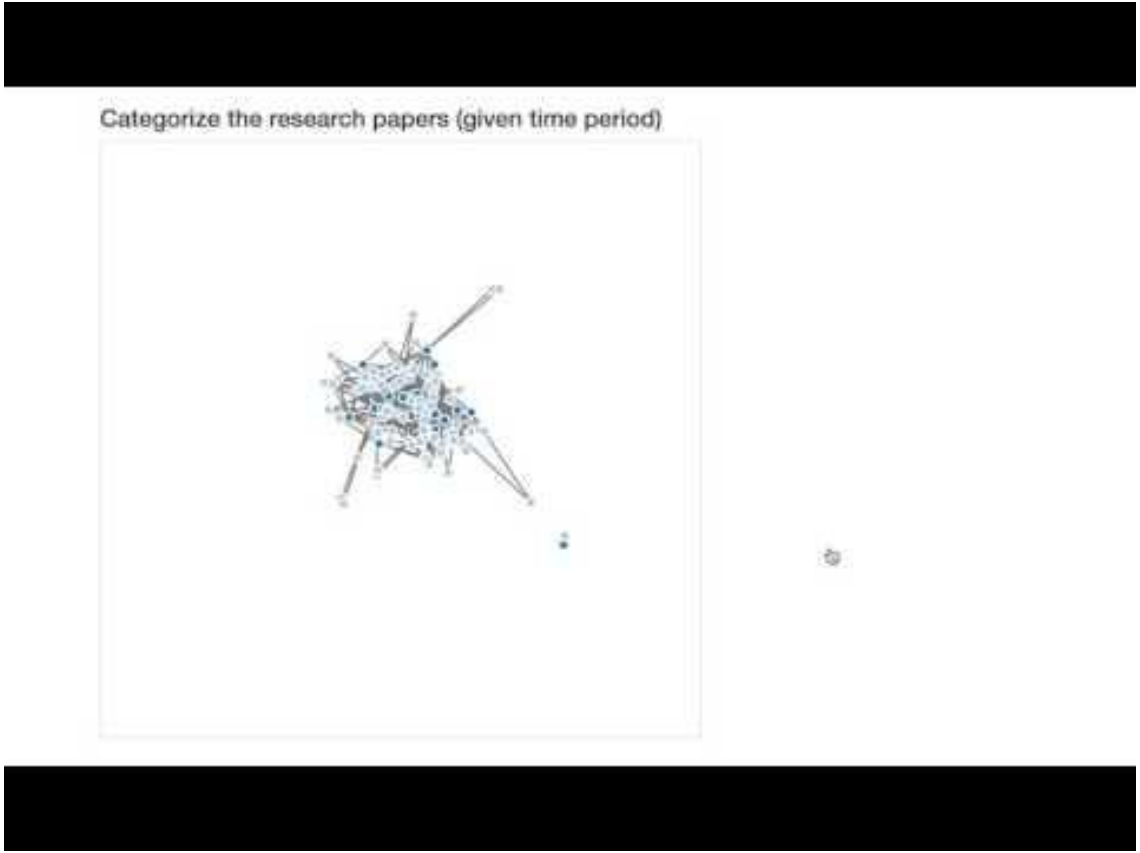


# System Implementation

- We use D3 to present our data and Bootstrap for the look and feel.
- We have a welcome page that shows all system features while allowing user to configure parameters.
- Most of our webpages are static htmls. However for those that need dynamic content, we use thymeleaf engine to generate the web page.



# Demo



# Experiment/ Analysis

**MySQL vs. Neo4j on a Large-Scale Graph Traversal**

**queries & user system**

*Inner Join vs Neo4j natural graph processing*

***e.g. prove the small world theory***

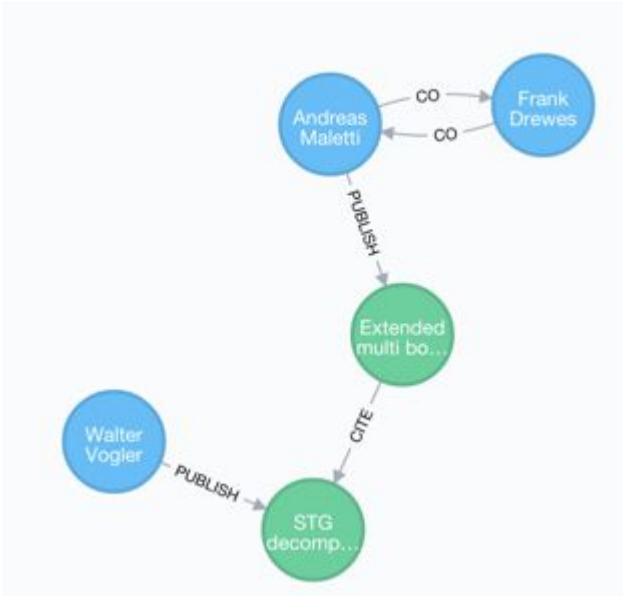
```
CREATE TABLE graph (
  outV INT NOT NULL,
  inV INT NOT NULL
);
CREATE INDEX outV_index USING BTREE ON graph (outV);
CREATE INDEX inV_index USING BTREE ON graph (inV);
```

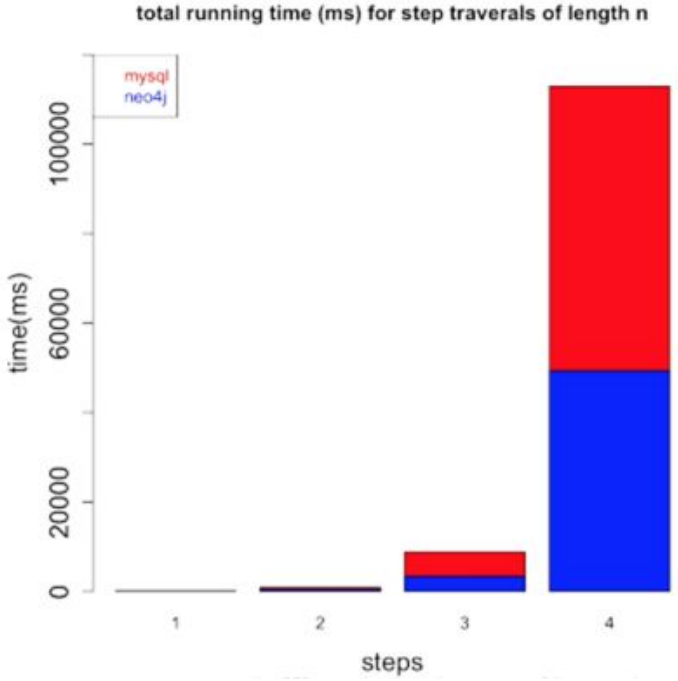
**query: SELECT a.name, b.name, c.name, d.name, e.name  From graph as a, graph as b, graph as c, graph as d where a.inV= b.outV and b.inV = c.outV and c.inV=d.outV and d.intV = e.outV and a.name = “Walter Vogler” and e. name=”Frank Drewis”**

# Experiment/ Analysis

**neo4j**

**query : match (a:Author {name: "Walter Vogler"}), (b:Author {name:"Frank Drewes"}), p = shortestPath((a)-[*..5]-(b)) return p**



# Experiment/ Analysis

**Reason :** Unlike relational databases, Neo4j stores interconnected master data that is neither linear nor purely hierarchical. Neo4j's native graph storing makes it easier to decipher your data by not forcing intermediate indexing at every turn.

**Conclusion:** Given a traversal of an artificial graph with natural statistics, the graph database Neo4j is more optimal than the relational database MySQL.

# Future Work

- Improve the Community System in the backend to enable following other authors / users.
- Build the  Recommendation System in the backend to enable recommendation based on user's recent search.
- Build the Custom Filter System to filter results based on keywords given by users.
- Improve the analysis function and user experience in the frontend.
- Improve the performance in the backend.

# Questions?

# Detailed Structure

- **What**: (what is the subsystem/project, what does it do, what is in it, what does it depend on)
- **Where**: (URLs, passwords, etc.) localhost:8080 (tomcat)
- **Who:** Who and how to contact for additional information (cell number, email)
- **How to install**: Maven + Neo4j
- **How to build:** mvn spring-boot:run -Drun.jvmArguments="-Dusername=neo4j -Dpassword=password"
- **How to run**: any modern browser
- **Known Issues, bugs, problems**: Performance can be an issue
- **Next steps**: More advanced search and recommendation mechanisms.